What is claimed is:

1. An apparatus for identifying software components, comprising:

a user interface means for obtaining object dependency and object usages information from a user;

a means of defining dependency weights for calculating weights of inter-object dependency based on the object dependency and the usages information;

a means of generating an object dependency network for representing degrees of object importance and inter-object dependency by using the dependency weights; and

a means of identifying software components for controlling the component identification process by using the object dependency network and the threshold values inputted by a user.

2. The apparatus of identifying software components as recited in claim 1, wherein the means of defining dependency weights considers not only the structural object dependency in the object model, but also the accumulated usages information among objects in the sequence diagrams of use cases in consideration of the importance weights of use cases.

3. The apparatus of identifying software components as recited in claim 1, wherein the means of generating the object dependency network represents dependency degrees (DD) among

objects, which are calculated by using structural dependency and the object usages information, and the importance degrees (ID) of each objects, which are calculated by summarizing the dependency degrees of connected objects.

4. The apparatus of identifying software components as recited in claim 1, wherein the means of identifying software components performs a clustering for grouping highly related objects on the object dependency network by considering the degrees of the object importance and the object dependency.

5. A method for identifying software components, comprising the steps of:

a) generating a use case & object dependency graph by using an object model and object usages information;

b) calculating a weight of each inter-object dependency and calculating a weight of object importance for each object by accumulating the dependency weights of connected objects;

c) determining seed objects, each of which has a greater importance value than a predetermined threshold and assigning each of the seed objects to a component; and

d) performing a navigation process to a non-navigated object in order to add the non-navigated object to the component, wherein an inter-object dependency value of the non-navigated object is larger than the predetermined threshold.

6. The method as recited in claim 5, wherein the step d) includes the steps of:

d1) setting initial conditions of the components for the navigation of an object;

d2) determining whether there is a component of which an object can be navigated or not;

d3) if there is the component, determining whether a non-included object of which the dependency value on the component is greater than a predetermined threshold exists in the component or not, if not, terminating the navigation process; and

d4) if there exists the non-included object in the component, adding the non-included object into the component and going back to the step d2), if not, setting Done[i] to "true", terminating the navigation process and going back to the step d2).

7. The method as recited in claim 5, wherein the step b) includes the steps of:

b) calculating a weight of each inter-object dependency and calculating a weight of object importance for each object by accumulating the dependency weights of connected objects;

b1) describing the dependency weights of inter-object dependency and the weights of object importance by positive real values with considering not only the structural object dependency in the object model, but also the accumulated usages information among objects in the sequence diagrams of

use cases;

b2) calculating a weight of the object importance for each object based on the dependency weights of inter-object dependency.

8. The method as recited in claim 5, wherein the step d) includes the step of performing a clustering for grouping highly related objects on the object dependency network by considering the degrees of the object importance and the object dependency.

9. A computer readable recording medium storing instructions for executing a method for identifying software components, which can be applied to an apparatus of identifying software component having a mass-storage processor, the component identification method comprising the steps of:

a) generating a use case & object dependency graph by using an object model and object usages information;

b) calculating a weight of each inter-object dependency and calculating a weight of object importance for each object by accumulating the dependency weights of connected objects;

c) determining seed objects, each of which has a greater importance value than a predetermined threshold and assigning each of the seed objects to a component; and

d) performing a navigation process to a non-navigated object in order to add the non-navigated object to the component, wherein an inter-object dependency value of the

non-navigated object is larger than the predetermined threshold.